REMARKS

Claims 1, 2, 4-41, 43-71 and 73-90 are pending in the application. Reconsideration is respectfully requested in light of the following remarks.

Section 103(a) Rejection:

The Examiner rejected claims 12-14, 20-39, 50-52, 54-70, 78 and 81-90 under 35 U.S.C. § 103(a) as being unpatentable over Johnson ("XML JavaBeans Integration, Part 3", 7/1999) in view of Allen (U.S. Patent 6,658,625). Applicants respectfully traverse this rejection for at least the following reasons.

Regarding claim 12, contrary to the Examiner's contention, **Johnson in view of Allen fails to teach or suggest a virtual machine receiving a data representation language representation of a first computer programming language object from a first process; a decompilation process of the virtual machine generating the first object from the data representation language representation of the first object, wherein the first object is an instance of a class in the computer programming language; and the decompilation process of the virtual machine providing the first object to a second process executing within the virtual machine**.

Johnson teaches an XMLJavaBeans class intended to provide a single process the ability to transform a JavaBean in memory into an XML document and vice versa. Allen teaches a method for generic data conversion using an XML-based data description including an XML parser to read and validate the data descriptions.

The Examiner admits that Johnson does not teach a decompilation process providing the first object to a second object executing within the virtual machine, but relies on Allen, citing col. 3, lines 12-24, and arguing, "Allen teaches it was known in the pertinent art, at the time applicant's invention was made, to reconstitute the object state in a distributed system" (Final Action, dated Feb. 21, 2008, p. 3). **However, the**

Examiner's interpretation of Allen is incorrect. Allen does not, neither at the cited passages nor elsewhere, whether considered singly or in combination with Johnson, teach that it was known to "reconstitute the object state in a distributed system." Instead, Allen teaches generic data conversion using a "data description." Allen's data description is a description of data format, and is not, nor does Allen suggest, a data representation language representation of a computer programming language object, as recited in Applicants' claim. Instead, Allen's data description is an XML-based description of various data formats. Allen's data description, which the Examiner incorrectly relies on as a data representation language representation of a computer programming language object, is clearly not a representation of an computer programming language object. Allen does not describe the data description as a data representation language representation of a computer programming language object. Instead, Allen teaches "a data description that has configurable data definitions that … can allow the data type, character set, location, and length of data elements in the data stream or file to be easily modified" (Allen, col. 3, lines 12-24). Further, Allen describes his data definition as "preferably describ[ing] the input and output data sent to and received from server program l95 and the information necessary to call server program l95" (Allen, col. 8, lines).

In his Response to Arguments, the Examiner argues that Allen teaches "a conversion from XML to an object" arguing that Allen's hash table representation of the data description is an object and citing col. 12, lines 1-20. However, even if Allen's hash table can be considered an object, the Examiner fails to take into account the fact that Applicants' claim does not merely recite, "a conversion from XML to an object." Instead, Applicants' claim recites, in part, "generating the first object **from the data representation language representation *of the first object***." In other words, the Allen's generic conversion of an XML-based data description to a hash table representation does not teach or suggest the specific limitation of generating an object *from* a data representation language representation *of that object*. As shown above, Allen's data definition is a *description* of data format, not a representation of a computer programming language object.

In fact, Allen makes clear that his data description is not a data representation language representation of an object. At col. 6, line 61 to col. 7, line 12, Allen states:

> "… a software engineer will now the data that needs to be converted from one format or type to another format or type. The data may be very complex. In particular, arrays, structures, or arrays of structures may be contained in the data. Each data element must be located in the data and converted, if necessary, from its current format or type into the correct format or type for the requesting computer. With the current invention, instead of hard-coding a program to find an convert all of these data elements, a software engineer can create a data description that describes the data elements and their interrelationships."

Thus, Allen's data description is clearly a document that describes how data is formatted in various formats and provides information for converting data between formats. Allen's data description is clearly not a data representation language representation of a computer programming language object, as recited in Applicants' claim.

In one example, Allen describes that when data is received from a server, "the format of the data is described by the data description" and that the data converter "uses the data description to convert the data from EBCDIC to Unicode" (Allen, col. 8, lines48-67). Thus, the data conversion relied on by the Examiner does not involve converting the data description, which the Examiner incorrectly equates with a data representation language representation of an object, but rather involved using the data description to convert other data received from a server.

Furthermore, as noted above, Johnson in view of Allen fails to teach or suggest receiving providing an object (generated from a data representation language representation of the object) ***to a second process*** **(i.e., a different process than the first process from which the data representation language representation of a first computer programming language object was received)**. Johnson's XMLJavaBeans class is clearly configured to return the results (e.g., the object generated from an XML document) to the same method, and hence to the same process, that called the

XMLJavaBeans interface (See, e.g., XMLBeanReader method definition, page 9). Allen's data converter and XML parser are directed to using PCML data descriptions to aid in the automatic conversion of data among different "multiple releases of the server using the same data description" and to "call the server program with the correct, converted parameters in the correct order" (Allen, col. 3, lines 7-24).

Allen fails to mention sending an object to a second (different) process. Allen's teaching regarding sending converter parameter data when calling server APIs does not teach or suggest the specific limitation of providing an object, generated from a data representation language representation of the object, to a second process. Allen describes generic data conversion and the use of traditional parameter data when calling server APIs. While Allen's converted data may be sent over a network to a server, Allen's data is sent across in various Server API formats, which do not include, and are not described as including, data representation language representations of object. Allen use of data representation languages (e.g., PCML) is limited to data descriptions, not representations of object. Nor does Allen teach the use of computer programming objects generated from data representation language representations of objects.

In the Response to Arguments, the Examiner asserts that Applicants' "claim does not recite that the second process [is] a 'different process than the first process from which the data representation language representation of a first computer programming language object was received'" (Final Action, p. 10). Applicants' respectfully disagree. Applicants' claim recites, in part, a virtual machine receiving a data representation language representation of a first computer programming language object "**from** a ***first*** process" and the decompilation process providing an object (generated from the data representation language representation of the object) "**to** a ***second*** process executing within the virtual machine." Thus, the plain meaning of Applicants' claim language recites a first process and a second process. The claim clearly recites *two* processes – a first process and a second process. The claim also recites receiving a representation **from** a first process and providing an object **to** a second process. If there was only one process involved, the claim would not have referred to both a first process and a second process.

Applicants' could have refrained from reciting the *second* process had they desired, but instead, claim 12 recites both a first process and a second process. The Examiner cannot ignore the plain meaning of Applicants' claim language.

Moreover, the Examiner has not provided a valid reason for modifying Johnson in view of Allen. The Examiner states that it would have been obvious to modify Johnson's system because "one having ordinary skill in the art would be motivated to reproduce object state in the distributed system" (Final Action, pp. 3-4). The Examiner's broad conclusory statement that Allen teaches that it was well known to "reconstitute object state" does not address the specific limitation and language of Applicants' claim. Desiring to "reproduce object state" does not provide a reason to include the generic data conversion system of Allen into Johnson's XMLJavaBeans. Firstly of all, Allen is not concerned with, nor teach anything regarding, reproducing object state in a distributed system. Instead, Allen is concerned with automatic generic data conversion to support the calling of different releases of server APIs. Additionally, the Examiner's stated opinion that one "would be motivated to reproduce object state" does not provide an actual, valid, reason why one of ordinary skill would be motivated to combine Johnson and Allen. The Examiner has simply stated his own opinion that one would be motivated, but fails to actually provide any reason as to why one would include the generic data conversion system of Allen into the XMLJavaBean system of Johnson. That the Examiner believes one would be motivated does not provide such a reason and is not supported by any actual evidence of record.

Additionally, the Examiner's combination of Johnson and Allen would not result in a system that included a virtual machine receiving a data representation language representation of a first computer programming language object from a first process; a decompilation process of the virtual machine generating the first object from the data representation language representation of the first object; and providing the first object to a second process executing within the virtual machine. A server sending or receiving Allen's converted data is clearly not a second process executing **within the virtual**

machine, as recited in Applicants' claim. The Examiner has failed to response to this argument when presented previously.

To establish a *prima facie* obviousness of a claimed invention, all claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d 981, 180 U.S.P.Q. 580 (C.C.P.A. 1974), MPEP 2143.03. More specifically, "All words in a claim must be considered in judging the patentability of that claim against the prior art" *In re Wilson*, 424 f.2d 1382, 1385,165 USPQ 494, 496 (CCPA 1970). As shown above, the cited art does not teach or suggest all limitations of Applicants' claims. As such, the Examiner has failed to provide a *prima facie* rejection of claim 12.

Thus, for at least the reasons above, the rejection of claim 12 is not supported by the cited art and removal thereof is respectfully requested. Similar remarks also apply to claim 78.

Regarding claim 25, Johnson in view of Allen fails to teach or suggest, generating a message in a data representation language, where the message includes a data representation language representation of the object, as recited in Applicants' claim. The Examiner relies on Allen, citing col. 3, lines 12-14 and asserting, "Allen teaches generating XML message of an object and providing such XML representation to another process" (Final Action, p. 6). Applicants respectfully disagree with the Examiner's interpretation of Allen.

As argued previously, Allen does not, even if combined with Johnson, teach generating an XML message of an object and providing that XML message to another process, as the Examiner asserts. In fact, Allen fails to teach generating any message in a data representation language that includes a data representation language representation of the object, which is apparently what the Examiner is arguing. Instead, as noted above, Allen teaches using XML-based (e.g., PCML) data descriptions to convert other data – not the XML data description – between data formats (Allen, col. 6, lines 35-41). Allen uses XML to describe data formats, **not for formatted messages including data**

representation language representations of objects. Nowhere does Allen describe an XML message including a data representation language representation of object. Instead, Allen teaches using the XML data descriptions to convert data into a format suitable for a particular release of a server API. For instance, Allen describes how the XML data definitions may specify both input and output parameters for particular server functions to be called (Allen, col. 6, lines 40-47; col. 9, lines 30-47).

As noted above, regarding claim 12, Allen teaches converting data received from a server (or being sent to a server) using an XML-based data description, but does not describe any data representation language representations of objects and clearly fails to describe , data representation language representations of objects in XML messages, contrary to the Examiner's assertions.

In the response to arguments, the Examiner erroneously asserts, "In Allen, the data conversion from/to XML is used to send and/or receive[] data from a server .. in XML", citing col. 6, lines 25-34). **However, the actual teachings of Allen, even in combination with Johnson, do not support the Examiner's assertion.** For example, at col. 6, lines 25-34, Allen fails to mention anything about using XML to send or receive data from a server, as suggested by the Examiner. Instead, Allen teaches, both at the cited passage and elsewhere, that XML-based data definitions are used to convert data sent to and received from a server, but does not teach that the data is sent to or received from the server *in XML*, as the Examiner suggests. Instead, *the entire point* of Allen's system is to use XML data descriptions to convert data in various formats. Nowhere does Allen mention that the data being converted is sent or received in XML, as the Examiner incorrectly asserts. The Examiner is clearly speculating regarding the detailed workings of Allen's system, which is clearly improper.

The Examiner also asserts in the Response to Arguments, "Allen teaches generating XML message of an object and providing such XML representation to another process in a distribut[ed] system", citing col. 3, lines 12-24). However, the cited passage, nor any other section of Allen, even in view of Johnson, does not teach or suggest,

"generating XML message of an object," as the Examiner contends. For instance, at the cited passage, Allen describes a generic data converter that uses XML-based data descriptions of data formats (not representations of objects, as the Examiner erroneously contends) to convert data sent and received from a server. Allen does not teach or suggest generating an XML message of an object as the Examiner asserts.

Furthermore, the Examiner's combination of cited art would not result in a system including generating messages in a data representation language including a data representation language representation of an object. For instance, incorporating Allen's XML data descriptions and generic data conversion into Johnson's XML JavaBeans system would include Johnson's ability to convert JavaBeans to and from XML, and would be able to use XML-based data descriptions when converting data. However, since, as shown above, neither Johnson nor Allen mentions generating a message in a data representation language or about including a data representation language representation of an object in such a message, no system resulting from a combination of Johnson and Allen would include such functionality.

Moreover, the Examiner has not provided a valid reason for modifying Johnson in view of Allen. The Examiner states that it would have been obvious to modify Johnson's system because "one having ordinary skill in the art would be motivated to reproduce object state in the distributed system" (Final Action, p. 6). However, desiring to "reproduce object state" would not motivate one to include the generic data conversion system of Allen into Johnson's XMLJavaBeans. Firstly of all, Allen is not concerned with, nor teach anything regarding, reproducing object state in a distributed system. Instead, Allen is concerned with automatic generic data conversion to support the calling of different releases of server APIs. Additionally, the Examiner's stated opinion that one "would be motivated to reproduce object state" does not provide an actual, valid, reason why one of ordinary skill would be motivated to combine Johnson and Allen. The Examiner has simply stated that one would be motivated, but fails to actually provide any reason as to why one would include the generic data conversion system of Allen into the XMLJavaBean system of Johnson. That the Examiner believes one would be motivated

does not provide such a reason and is not supported by any actual evidence of record. The Examiner has not responded to this argument in the Response to Arguments of the Final Action.

Thus, the rejection of claim 25 is not supported by the cited art and removal thereof is respectfully requested. Similar remarks also apply to claims 62 and 84.

Regarding claim 34, Johnson in view of Allen fails to teach or suggest a first process receiving a message in a data representation language from a second process, where the message includes information representing a computer programming language object. The Examiner relies on Allen, citing col. 3, lines 12-24. However, as shown above regarding the rejection of claim 25, Allen, even if combined with Johnson, does not teach or suggest message in a data representation language or including information representing a computer programming language object in such a message, contrary to the Examiner's assertions.

Instead, Allen teaches using XML-based (e.g., PCML) data descriptions to support automatic data conversions among various releases of server APIs. For instance, Allen's system reads and parses a XML-based data description that describes various data elements, such as input and output parameters to server API functions, so that the data can be converted to a suitable format for the particular release of the server API being called (Allen, col. 6, lines 35-41). Allen's data descriptions are simply not messages in a data representation language. Instead, Allen's data descriptions are stored in an XML file that is read and parsed (not send as a message) by Allen's system.

In the Response to Arguments, the Examiner argues, "in Allen's distributed system where API layer provides interfaces to clients and services, the interfaces are for message communicating between objects" and that "XML data are sen[t]/received as a message in the distributed system," citing col, 2, lines 23-26 and col. 7, lines 50-61. However, the Examiner's cited passages do not support the Examiner's argument (nor indeed does any part of Allen, even in view of Johnson). Allen's description of the API

relied on by the Examiner does not include any mention of XML data or about the format of any messages used in conjunction with the API. Applicants are not arguing that Allen's does not or cannot use messages. Applicants are arguing that Allen, even if combine with Johnson, does not teach or suggest XML messages including data representation language representations of object, as the Examiner erroneously contends and relies on.

The Examiner is confusing the fact that Allen uses an XML-based language (PCML) for data descriptions with data representation language representations of objects as recited in Applicants' claim. As argued above and previously, Allen's data descriptions are not representations of objects.

As shown above, the cited art does not teach or suggest all the limitations of Applicants' claim. Thus, the rejection of claim 34 is not supported by the cited art and removal thereof is respectfully requested. Similar remarks also apply to claim 50.

Applicants also assert that the rejection of numerous ones of the dependent claims is further unsupported by the teachings of the cited art. However, since the rejection of the independent claims has been shown to be unsupported by the cited art, a further discussion of the rejection of the dependent claims is not necessary at this time.

Allowed Claims:

Claims 1, 2, 4-11, 40, 41, 43-49, 71 and 73-77 have been allowed by the Examiner. Applicants thank the Examiner for consideration of these claims.

Allowable Subject Matter:

Claims 15-19, 53 and 79-80 were objected to as being dependent upon a rejected base claim, but would be allowable if rewritten in independent form. Applicants thank the Examiner for consideration of claims 15-19, 53 and 79-80, but assert that these claims

are allowable as current written for at least the reasons given above regarding their respective, independent claims.

CONCLUSION

Applicants submit the application is in condition for allowance, and notice to that effect is respectfully requested.

If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5181-72000/RCK.

Respectfully submitted,

/Robert C. Kowert/
Robert C. Kowert, Reg. #39,255
Attorney for Applicants

Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8850

Date: April 20, 2008